BLACK ROT DISEASE DETECTION IN GRAPE PLANT USING COLOUR BASED SEGMENTATION AND MACHINE LEARNING

A Mini Project Report

submitted by

AMRITHA U(MES20MCA-2006)

to

the APJ Abdul Kalam Technological University
in partial fulfillment of the requirements for the award of the Degree

of

Master of Computer Applications

Department of Computer Applications

MES College of Engineering
Kuttippuram, Malappuram - 679 582

February 2022

DECLARATION

I undersigned hereby declare that the project report **BLACK ROT DISEASE DETECTION IN GRAPE PLANT USING COLOUR BASED SEGMENTATION AND MACHINE LEARNING**, submitted for partial fulfillment of the requirements for the award of degree of Master of Computer Applications of the APJ Abdul Kalam Technological University, Kerala, is a bona fide work done by me under supervision of Mr.HYDERALI K, Assistant Professor, Department of Computer Applications. This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Place:KUTTIPURAM

Date:28-02-2022

AMRITHA U(MES20MCA-2006)

DEPARTMENT OF COMPUTER APPLICATIONS
MES COLLEGE OF ENGINEERING, KUTTIPPURAM

CERTIFICATE

This is to certify that the report entitled **BLACK ROT DISEASE DETECTION IN GRAPE PLANT USING COLOUR BASED SEGMENTATION AND MACHINE LEARNING** is a bona fide record of the Mini Project work carried out by **AMRITHA U(MES20MCA-2006)** submitted to the APJ Abdul Kalam Technological University, in partial fulfillment of the requirements for the award of the Master of Computer Applications, under my guidance and supervision. This report in any form has not been submitted to any other University or Institution for any purpose.

Internal Supervisor(s) External Supervisor(s)

Head Of The Department

Acknowledgements

My endeavor stands incomplete without dedicating our gratitude to a few people who have contributed towards the successful completion of my project.I pay our gratitude to the Almighty for his invisible help and blessing for the fulfillment of this work.At the outset I express our heartful thanks to our Head of the Department and our guide Mr.Hyderali K for his valuable guidance and supervision. I take this opportunity to express our profound gratitude to Ms.Priya J D as my project coordinator for her valuable support, timely advise and strict schedules to complete our project. I also grateful to all our teaching and non-teaching staff for their encouragement, guidance and whole-hearted support. Last but not least, I gratefully indebted to our family and friends, who gave us a precious help in doing our project.

AMRITHA U(MES20MCA-2006)

Abstract

Black Rot is a fungal disease which affects the yield as well as the wine quality and can also cause complete crop loss. It can be identified as brown/tan coloured circular spots/lesions distributed unevenly on the leaf of the plant. A proper detection of the disease is required which can be further helpful in taking active measures like Spraying of Fungicides, Pruning, etc. can be done on time. The PlantVillage Dataset is used, which contains images of grape plant leaves affected from Block Rot Disease as well as the pictures of healthy leaves. HSV and L*a*b* colour models are used for the segmentation purposes. The healthy part and the diseased part of the leaves are separated using colour- based techniques and the features are stored for each leaf. The color of diseased part is very much different from the healthy part of the leaves which makes it easier to detect the disease on the basis of color. The machine learning is done using the K-Nearest Neighbour Classifier and the results are analysed on different clusters of K-NN. The highest accuracy achieved is 94.1

Keywords:—Disease Detection, Pattern Recognition, K-NN, HSV, Colour Based Detection, Grape Plant, Image Processing, L*a*b*, K-means Clustering, PlantVillage Dataset, Supervised Learning, Machine Learning.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

This paper discusses about a model which can be used for early detection of Black Rot disease in Grape Vines. It uses a colour-based segmentation and detection of the disease. The two-colour models i.e. HSV and L*a*b* are used to segment the affected areas from the pictures of diseased leaves. The system is developed using machine learning algorithms which collects the features of the disease and get trained by that data. The trained feature vectors are then compared with the feature vectors to be tested. The accuracy is computed on the basis of those comparisons. K-Nearest Neighbour (K-NN) is used to classify the diseased and non-diseased (healthy) leaves. This data can be used for further analysis and can be used to automate the early disease detection procedure using robotics,etc.

a). Pre-processing phase:In the first phase, the plant leaf image is given which is converted to grayscale. This is actual step in which the features are extracted from the image that has undergone segmentation. Feature extraction is the means through which one trains the training set data which is to be used in the next step.

b).Image Segmentation: The Image segmentation technique is applied which will segment the image based on their properties. Acquiring the image and splitting it into several regions (or segments) which are further used individually for extracting out the image features is termed as the segmentation. In this work, the region based k-mean segmentations technique is applied for the image segmentation.

c).K-mean clustering: The approach that we are implementing is k-mean clustering algorithm

for segmentation. It is one of the popular methods used for segmentation. In this we divide the image into various clusters i.e. divides a set of data into specific number of groups. Data is classified into k number of disjoint sets. A k-centroid is determined and then each point which has least distance from the centroid is taken into consideration. There are several ways of defining the distance of nearest centroid, one such method is Euclidean distance. Euclidean distance is calculated consecutively for each data point and the data point having the minimum distance is assigned to the cluster. These minimum points are summed up to get a centroid. This is how k-mean clustering works.

1.1.1 Motivation

The identification of plant disease is the premise of the prevention of plant disease efficiently and precisely in the complex environment. With the rapid development of the smart farming, the identification of plant disease becomes digitalized and data-driven, enabling advanced decision support, smart analyses, and planning.Through this we can easily identify leaf disease and people can use this app from anywhere and get fast result.

1.2 Objective

The objective of this paper is ,There are many pathogens which can cause disease in the plants, one of them is fungus. The most common fungal disease among grape plants is Black Rot disease caused by a fungus named as Guignardiabidwellii. Black Rot Disease was observed in the Grape Plants which were growing in the places with high humid conditions. The pathogens release some toxic compounds on the leaves of the plant which results in necrotic (causing the effected tissue's death) spots which appeared to be brownish-tan coloured. These spots can be seen on all green regions of the leaf. This disease can also become an epidemic sometimes resulting in yield loss from 5 percentage to 80 percentage. Relatively small, brown circular lesions develop on infected leaves and within a few days tiny black spherical fruiting bodies protrude from them. Elongated black lesions on the petiole may eventually girdle these organs, causing the affected leaves to wilt. Shoot infection results in large black elliptical lesions. These lesions may contribute to breakage of shoots by wind, or in severe cases, may girdle

and kill young shoots altogether. This fungus bides its time. Most plants show very little signs of infection until its too late. They will look very healthy until fruit sets. Even flowering will be normal. Infection of the fruit is the most serious phase of the disease and may result in substantial economic loss. Infected berries first appear light or chocolate brown; it will have a spot that looks very round, like the eye of a bird. That spot will get larger and infect more of the fruit bunch and more of the plant. This creates masses of black pycnidia developing on the surface. Finally, infected berries shrivel and turn into hard black raisin-like bodies that are called mummies. Here we use support vector machine algorithm to predict Black Rot disease in grape plants by analysing leaf images.

1.3 Report Organization

The project report is divided into four sections. Section 2 describes literature survey. Section 3 describes the methodology used for implementing the project. Section 4 gives the results and discussions. Finally Section 5 gives the conclusion.

Chapter 2

Literature Survey

Rossi, V., Onesti, G., Legler, S.E has proposed the available knowledge on black-rot of grape was retrieved from literature, analyzed, and synthesized to develop a mechanistic model of the life cycle of the pathogen (Guignardia bidwelii) based on the systems analysis. Three life-cycle compartments were defined: (i) production and maturation of inoculum in overwintered sources (i.e., ascospores from pseudothecia and conidia from pycnidia in berry mummies and cane lesions); (ii) infection caused by ascospores and conidia; and (iii) disease onset and production of secondary inoculum. An analysis of published, quantitative information was conducted to develop a mechanistic model driven by weather and vine phenology; equations were developed for ascospore and conidial maturation in overwintered fruiting bodies, spore release and survival, infection occurrence and severity, incubation and latency periods, onset of lesions, production of pycnidia, and infectious periods. The model was then evaluated for its ability to represent the real system and its usefulness for understanding black-rot epidemics by using three typical epidemics. Finally, weaknesses in our knowledge are discussed. Additional research is needed concerning the influence of wetness duration and temperature on infection by ascospores, production dynamics of pycnidia and conidia in black-rot lesions, and the dynamics of conidia exudation from pycnidia.

M. R. Sosnowski, R. W. Emmett, W. F. Wilcox, T. J. Wicks, has proposed a drastic pruning strategy was developed to eradicate the fungal disease black rot (Guignardia bidwellii), which is exotic in Australia, from grapevines, while minimizing the economic cost of returning an affected vineyard to its previous quality and production levels. The protocol involved cutting off vines at the top of the trunk, removing debris from the ground beneath and between

vines, mulching the vineyard floor, removing low watershoots during vine regrowth and applying a targeted fungicide programme. The protocol was initially evaluated and consequently modified in Australia using an endemic grapevine disease, black spot or anthracnose (Elsinoe ampelina), as an analogous model system. Then, it was validated in a black-rot-infested vineyard in New York, USA. Following two seasons of disease-conducive weather conditions, no black rot was detected on treated vines, whereas leaf and fruit infections developed on the untreated control vines. These results confirmed the efficacy of the protocol for eradicating black rot from vineyards while allowing vines to return quickly to previous yield and quality levels without replanting. The protocol may have applicability to disease eradication protocols for other perennial crops as well. Evidence is also presented on the efficacy and potential pitfalls of burning infected grapevine material to eradicate E. ampelina.

Chapter 3

Methodology

3.1 Introduction

The methodology explained below uses 2 colour modelsHSV model and L*a*b* model,It involves the image to be converted into HSV format. Theleaf part is segmented from the image. Then, the diseasedpart is extracted using L*a*b* model in combination withK-means Adaptive Clustering. Gray-Scaling is applied onthe extracted healthy and diseased images and binarizationis done using Otsu's Global Threshold method. The pixel count is computed on the basis of white pixels in both cases (healthy as well as diseased). The classification is applied using supervised machine learning algorithm, K-Nearest Neighbour algorithm with its different Clusters.

3.2 Modules

The project is divided into 2 functional modules they are,

1.Expert System

• Login

• Add and manage dataset

• View farmers

• Chat with farmers

• Send notification

• View feedback

• Add fertilizer

2.Farmers

• Registration

• Login

• Upload leaf image View prediction result

• Chat with experts

• View notifications

• Send feedbacks

• View fertilizer

3.3 Developing Enviornment

-OPERATING SYSTEM : WINDOWS 10

-FRONT END : HTML, CSS, JAVASCRIPT -BACK END : MySQL

-Dataset: Plant Village Dataset from Kaggle website is used

-IDE : JetBrains PyCharm, Android studio

-TECHNOLOGY USED : PYTHON,JAVA

-FRAME WORK USED : Flask

3.4 Workflow

FEATURES EXTRACTION USING COLOR MODELS HSV L*A*B*

The RGB image in the dataset is loaded into the system and converted into HSV image. This is done to extract the saturation channel from the image. It signifies the amount of colour in the pixel. The Hue, Saturation and Intensity valuesare computed using the formulae 1-9 [21]. R, G B signifiesthe values for Red, Green and Blue channel values respectively for the pixel.

a) Input image

b) Image after conversion to HSV model

c) Extracted saturation channel

d) Labelled clusters by k-means clustering

e) Segmented leaf

f) Conversion to L*a*b model

h) Segmented unaffected part

Figure shows the process of feature extraction steps,

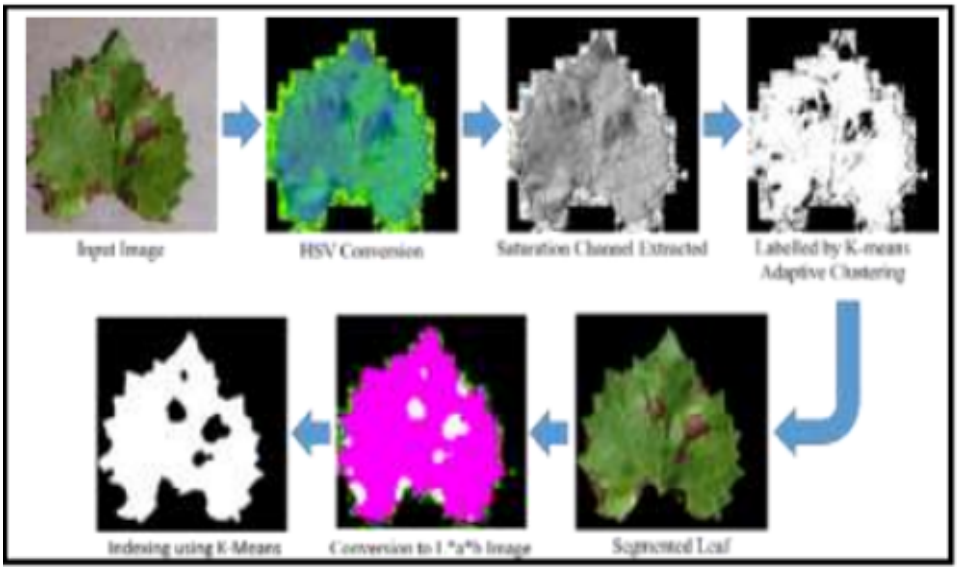

Figure 3.1: Feature extraction using HSV L*a*b

SEGMENTATION

Image Segmentation: The Image segmentation technique is applied which will segment the image based on their properties. Acquiring the image and splitting it into several regions (or segments) which are further used individually for extracting out the image features is termed as the segmentation. In this work, the region based k-mean segmentations technique is applied for the image segmentation.

a) Segmented green/unaffected area

b) Gray scaling

c) Binarization

d) Segmented brown/unaffected area

e) Gray scaling of unaffected area

f) Binarization of affected area

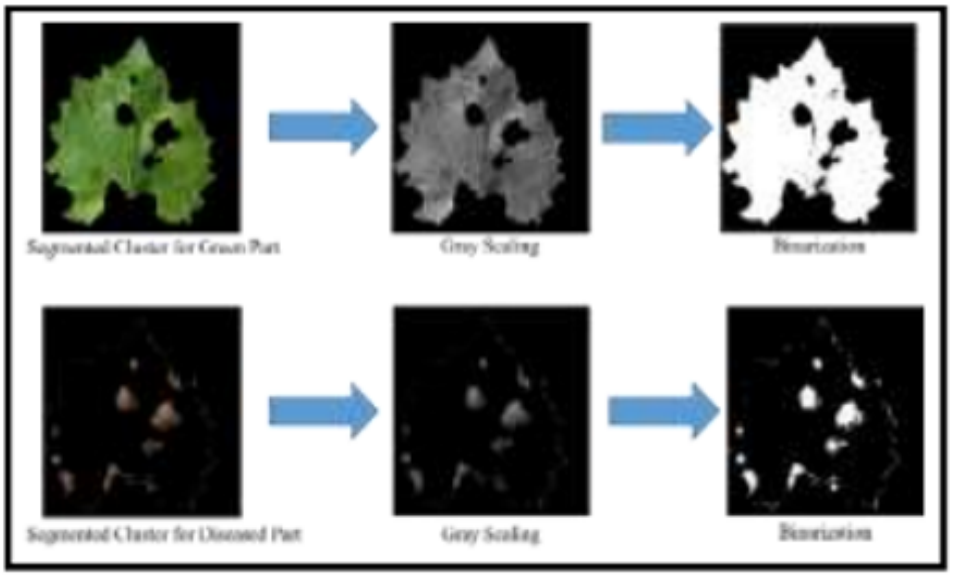

Figure 3.2: Segmentation

3.5 K-Nearest Neighbour(K-NN) Algorithm

Step 1 : Load the data

Step 2 : Divide the data into training data and test data

Step 3 : 1Initialize the value of k

Step 4 :Calculate the Euclidean distance between the test data and each of the training data

Step 5 :Find k number of nearest neighbors having minimum distance values

Step 6 :Get the most frequent class of these neighbors

Step 7 :This will be the predicted class

- **choice of k**

1. If k is small, the result will be affected by noisy data
2. If k is large, the algorithm will be computationally expensive
3. The best k value is somewhere between these two extremes
4. We can choose k as the square root of the number of training data
5. An alternative approach is to test several k values on a variety of test data sets and choose the one that delivers the best performance

3.6 USER STORY

A key component of agile software development is putting people first, and user-stories put actual end users at the center of the conversation. Stories use non-technical language to provide context for the development team and their efforts. After reading a user story, the team knows why they are building what they're building and what value it creates. A user story is a tool used in agile software development to capture a description of a software feature from an enduser perspective. The user story describes the type of user, what they want and why. A user story helps to create a simplified description of a requirement. User stories are one of the core components of an agile program. They help provide a user-focused framework for daily work which drives collaboration, creativity, and a better product overall.

UserStoryID	As a <type of user>	I want to	So that I can
1	Expert	login	login successful with correct username and password
2	Expert	Add & Manage Dataset	Add disease effected leaf Image to dataset, compare to uploaded image in here to predict disease
3	Expert	View farmer	View registered farmer details
4	Expert	Chat with farmers	Chat with farmers
5	Expert	Send notification	Send notification to user
6	Expert	View feedback	View user feedback
7	Expert	Add fertilizer	Add fertilizer details
8	User	Registration	User can register
9	User	Login	Login successful with correct username and password
10	User	Upload leaf image & view prediction result	Upload image and view result
11	User	Chat with expert	Chat with expert
12	User	View notification	View notification
13	User	Send feedback	Send feedback to expert
14	User	View fertilizer	View fertilizer

Figure 3.3: Userstory

3.7 PROJECT PLAN

A project plan that has a series of tasks laid out for the entire project, listing task durations, responsibility assignments, and dependencies. Plans are developed in this manner based on the assumption that the Project Manager, hopefully along with the team, can predict up front everything that will need to happen in the project, how long it will take, and who will be able to do it.

User Story ID	Task Name	Start Date	End Date	Days	Status
1	Sprint 1	26/12/2021	28/12/2021	2	Completed
2		29/12/2021	31/12/2021	3	Completed
3		03/12/2021	08/01/2022	5	Completed
4	Sprint 2	09/01/2022	16/01/2022	8	Completed
5		18/01/2022	22/01/2022	5	Completed
6	Sprint 3	23/01/2022	27/01/2022	5	Completed
7		30/01/2022	05/02/2022	7	Completed
8	Sprint 4	06/02/2022	10/02/2022	5	Completed
9		16/02/2022	20/02/2022	4	Completed

Figure 3.4: Project plan

3.8 PRODUCT BACKLOG

A product backlog is a list of the new features, changes to existing features, bug fixes, infrastructure changes or other activities that a team may deliver in order to achieve a specific outcome.The product backlog is the single authoritative source for things that a team works on. That means that nothing gets done that isn't on the product backlog. Conversely, the presence of a product backlog item on a product backlog does not guarantee that it will be delivered. It represents an option the team has for delivering a specific outcome rather than a commitment.It should be cheap and fast to add a product backlog item to the product backlog, and it should be equally as easy to remove a product backlog item that does not result in direct progress to achieving the desired outcome or enable progress toward the outcome. The Scrum Product Backlog is simply a list of all things that needs to be done within the project. It replaces the traditional requirements specification artifacts. These items can have a technical nature or can be user-centric e.g. in the form of user stories.

User Story ID	Priority <High/Medium/Low>	Size (Hours)	Sprint <#>	Status <Planned/In progress/Completed>	Release Date	Release Goal
1	Medium	2	1	Completed	8/01/2022	Table design
2	High	3		Completed	8/01/2022	Form design
3	High	5		Completed	8/01/2022	Basic coding
4	High	5	2	Completed	22/1/2022	collects the features of the disease
5	Medium	5		Completed	22/01/2022	Training the data
6	High	5	3	Completed	5/02/2022	classify different leaf images using SVM
7	Medium	5		Completed	17/02/2022	find Black Rot disease
8	Medium	5	4	Completed	20/02/2022	Testing data
9	High	5		Completed	20/02/2022	Output generation

Figure 3.5: Product backlog

3.9 SPRINT BACKLOG PLAN

The sprint backlog is a list of tasks identified by the Scrum team to be completed during the Scrum sprint. During the sprint planning meeting, the team selects some number of product backlog items, usually in the form of user stories, and identifies the tasks necessary to complete each user story. Most teams also estimate how many hours each task will take someone to complete.

Backlog item	Status and completion date	Original estimate in hours	Day1	Day2	Day3	Day4	Day5	Day6	Day7	Day8	Day9	Day10	Day11	Day12	Day13	Day14
User story#1,#2,#3			hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs
Table design	28/12/2021	2	1	1	0	0	0	0	0	0	0	0	0	0	0	0
Form design	31/12/2021	3	0		1	1	1	0	0	0	0	0	0	0	0	0
Basic coding	08/01/2022	5	0	0	0	0	0	1	1	1	1	1	0	0	0	0
User story #4,#5																
collects the features of the disease	16/01/2022	5	1	1	0	1	1	1	0	0	0	0	0	0	0	0
Training the data	22/01/2022	5	0	0	0	0	0	0	0	1	1	0	1	1	1	0
User story #6,#7																
classify different leaf images using SVM	27/01/2022	5	1	1	1	0	1	1	0	0	0	0	0	0	0	0
find Black Rot disease	05/02/2022	5	0	0	0	0	0	0	0	1	1	1	1	1	0	0
User story #8,#9																
Testing data	10/02/2022	5	1	1	1	1	1	0	0	0	0	0	0	0	0	0
Output generation	20/02/2022	5	0	0	0	0	0	0	2	2	2	0	0	0	0	0
Total		40	4	4	3	3	4	3	3	5	4	2	2	2	1	0

Figure 3.6: Sprint backlog plan

3.10 SPRINT ACTUAL

Actual sprint backlog is what adequate sprint planning is actually done by project team there may or may not be difference in planned sprint backlog.

Backlog item	Status and completion date	Original estimate in hours	Day1	Day2	Day3	Day4	Day5	Day6	Day7	Day8	Day9	Day10	Day11	Day12	Day13	Day14
User story#1,#2,#3			hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs
Table design	28/12/2021	2	1	1	0	0	0	0	0	0	0	0	0	0	0	0
Form design	31/12/2021	3	0	0	2	1	0	0	0	0	0	0	0	0	0	0
Basic coding	08/01/2022	5	0	0	0	0	0	1	1	1	2	0	0	0	0	0
User story #4,#5																
collects the features of the disease	22/01/2022	8	2	0	0	2	0	2	0	0	1	0	1	0	0	0
Training the data	22/01/2022	5	1	0	0	0	2	0	0	1	0	0	1	0	0	0
User story #6,#7																
classify different leaf images using SVM	5/02/2022	5	1	0	0	0	2	0	0	0	0	2	0	0	0	0
find Black Rot disease	17/02/2022	7	2	0	0	0	0	2	0	0	0	0	2	1	0	0
User story #8,#9																
Testing data	20/02/2022	5	2	0	0	0	0	0	0	2	0	0	0	0	1	0
Output generation	20/02/2022	4	0	0	0	0	0	0	2	2	0	0	0	0	0	0
Total		44	9	1	2	3	4	5	3	6	3	2	4	1	1	0

Figure 3.7: Sprint Actual

Chapter 4

Results and Discussions

4.1 Datasets

The images used in the study are accessed from a web- based repository and has monochrome background. The images are mainly of the Grape Vine Leaves.

A. Image Database

The images that are used for analysis are taken from the PlantVillage Grape Plant Dataset (Black Rot Disease) [20]. The database contained the images of leaves of grape plant which are commonly single leaves and the background was monochrome hence, it was easier to segment the leaf areas with no complex background. 400 images where used in total. Each image in the dataset was of 256x256 resolution. The healthy leaves have no spots and has flawless texture, on the other hand, diseased leaves are affected with brown spots or lesions.The data set used for the training and testing of the system was collected from **kaggle.com**.

4.2 Results

The image dataset contained 400 images, out of which, 250 images are used for training purpose and 150 images are used for testing purpose. The feature vectors made are then processed for classification phase. Support Vector machine is used for classification. The data is analyzed on K-NNs clusters.

In the existing technique, linear SVM is used for classification. The linear SVM only classify the data into two classed which is very inefficient and reduce accuracy of classification. In this

work, other classifier than SVM will be applied which classify data more than two classes. we applied KNN which will classify more than two type of classes. The image is given as input from which we need to detect the disease. The input image will be converted to the gray scale to apply GLCM algorithm for the feature extraction which will extract the textural features like energy, entropy etc. These input features are used for the plant disease detection.

4.3 SCREENSHOTS

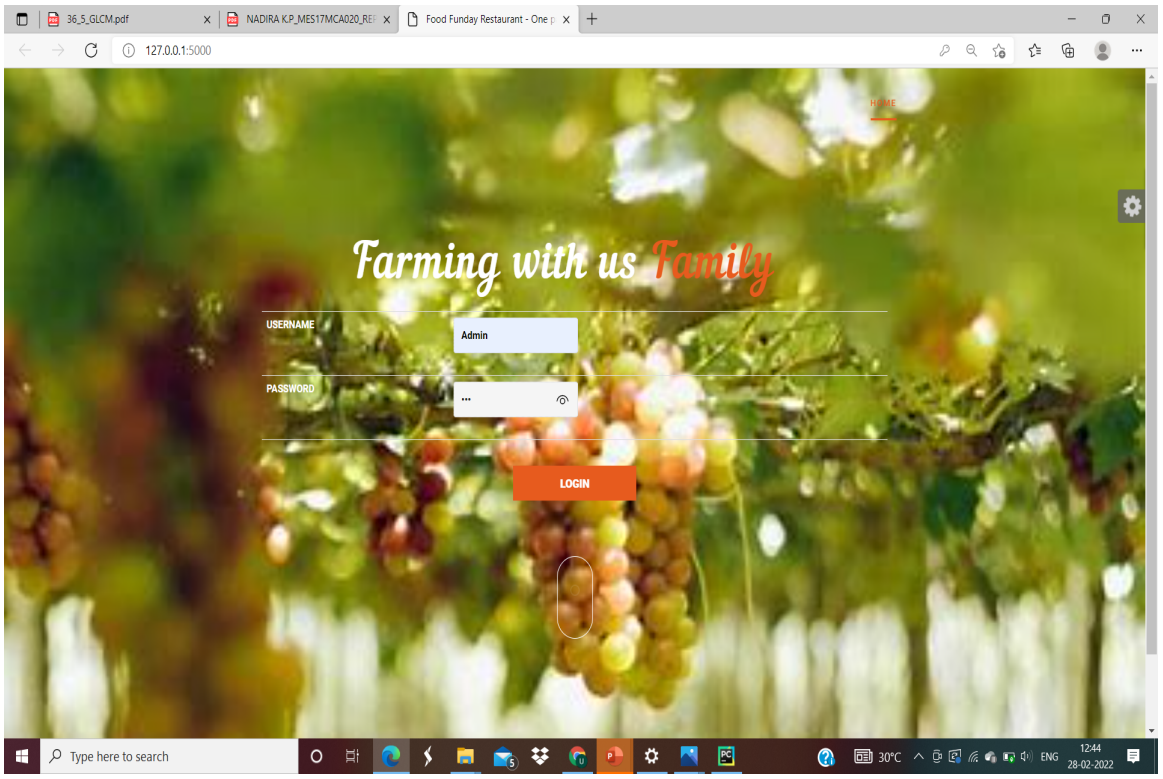

Figure 4.1: Screenshot

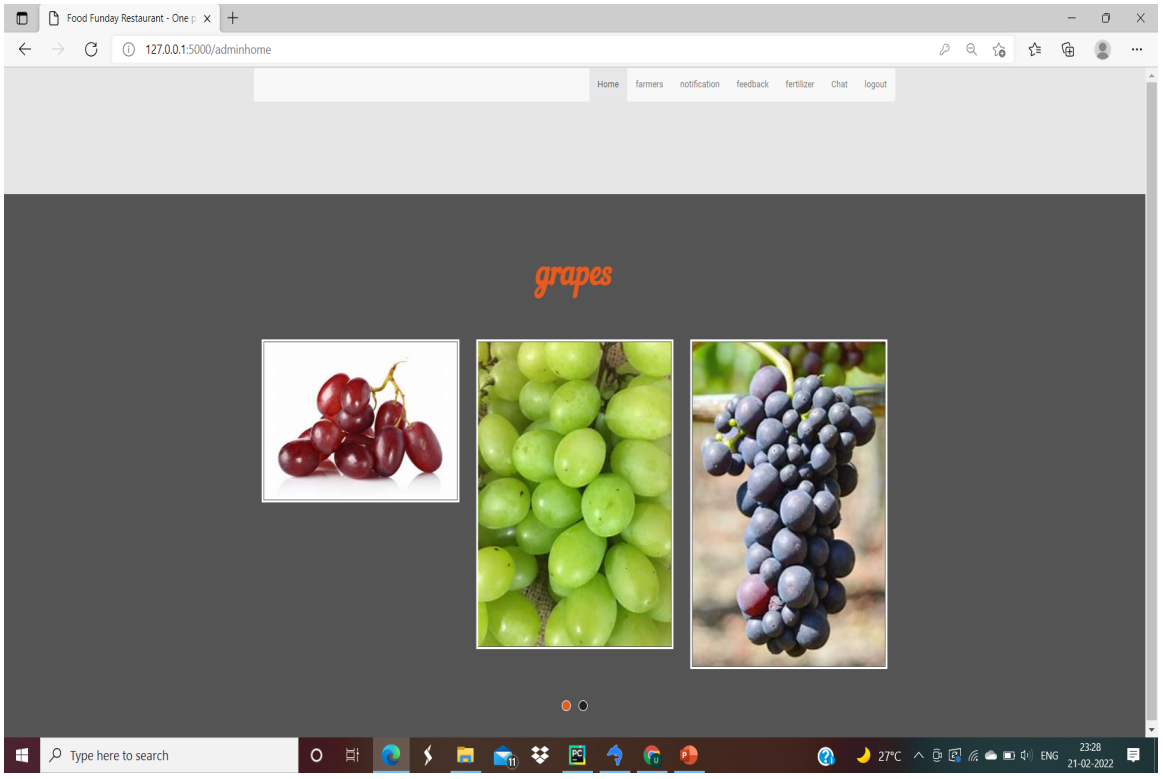

Figure 4.2: Screenshot

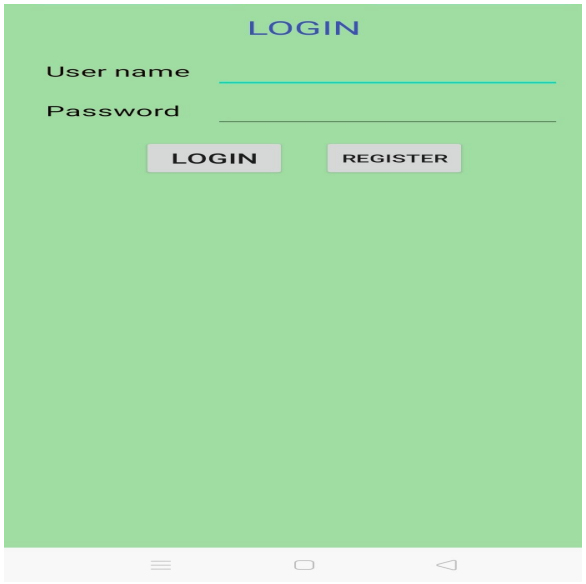

Figure 4.3: Screenshot

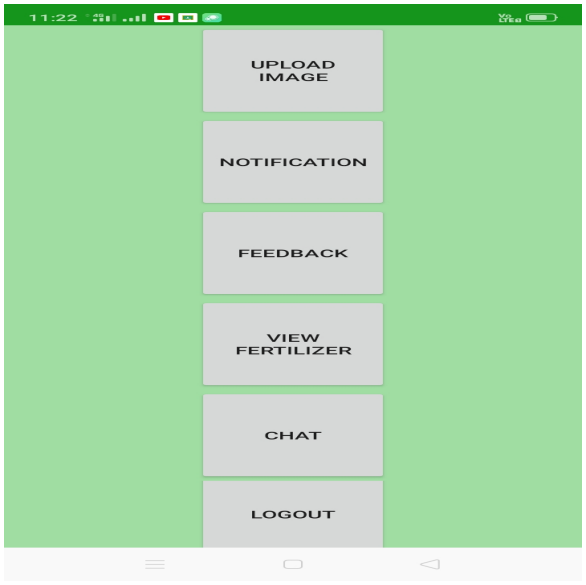

Figure 4.4: Screenshot

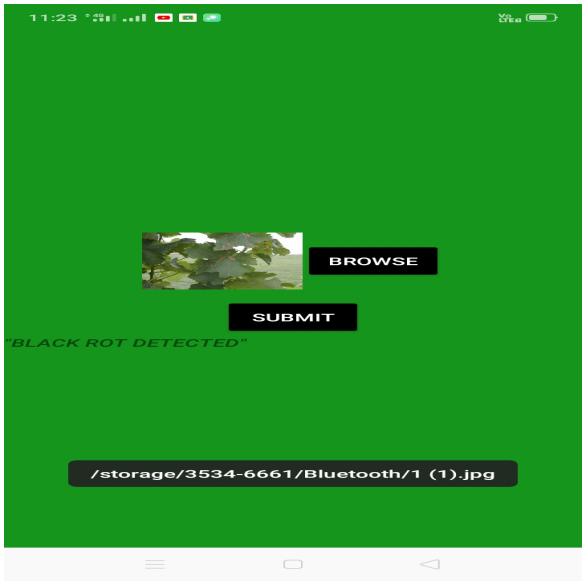

Figure 4.5: Screenshot

Chapter 5

Conclusions

The ststem mentioned 2 color models- HSV model and L*a*b* model can be used to segment the leaf from the background and diseased part from the leaf. The diseases in plants can be segmented/separated with the use of color-basedimage processing techniques. The color percentage ratio is proven to be a successful method in detecting the disease if the affected area has different color to the non-affected area. The system performs best with K-NN Clusters.
it is being concluded that plant disease detection required three main steps namely feature extraction, segmentation and classification. In the existing technique GLCM algorithm is used to extract the textural features. The k-mean clustering is applied to segment input images. The SVM classifier is replaced with the KNN classifier in the proposed work to classify data into multiple classes. The performance of proposed algorithm is tested in terms of accuracy and false positive rate with is increase up to 10 percent as compared to existing technique.

References

[1] **Rossi, V., Onesti, G., Legler, S.E** "Use of systems analysis to develop plant disease models based on literature data: grape black-rot as a case-study", European Journal of Plant Pathology, 141, Issue 3, March 2015, pp 427–444.

[2] **R.Pydipati,T.F.Burks,W.S.Lee,** "Identification of citrus disease using color texture features and discriminant analysis", Computers and Electronics in Agriculture, Volume 52, Issues 1–2, June 2006, pp 49-59.

[3] **Vijai Singh, A.K.Misra,** "Detection of plant leaf diseases using image segmentation and soft computing techniques", Information Processing in Agriculture, Volume 4, Issue 1, March 2017, pp 41-49.

Appendix

Source Code

```
\\webcode.py
  from flask import *

from src.dbconnection import *

app=Flask(__name__)
import functools

def login_required(func):
   @functools.wraps(func)
   def secure_function():
      if "lid" not in session:
         return redirect("/")
      return func()

   return secure_function
app.secret_key="qwe"
@app.route('/')
def login():
   return render_template('login.html')

@app.route('/logincode',methods=['post'])
def logincode():
   uname=request.form['textfield']
   pwd=request.form['textfield2']
   qry="SELECT * FROM login WHERE `username`=%s AND `password`=%s"
   val=(uname,pwd)
   res=selectone(qry,val)
   print(res[0])
   if res is None:
      return '''<script>alert("invalid");window.location="/"</script>'''
   else:
      if res[3]=='admin':
         session['lid']=res[0]
         return redirect('/adminhome')

      else:
         return '''<script>alert("invalid");window.location="/"</script>'''



@app.route('/addfertilizer',methods=['post'])
@login_required
```

Appendix

```
def addfertilizer():
    return render_template('addfertilizer.html')


@app.route('/addfertilizers',methods=['post'])
@login_required

def addfertilizers():
    name=request.form['textfield']
    des=request.form['textarea']
    uses=request.form['textarea2']

    qry="insert into fertilizer values(null,%s,%s,%s)"
    val=(name,des,uses)
    iud(qry,val)
    return '''<script>alert("fertilizer added");window.location="/fertilizer"</script>'''

@app.route('/adminhome')
@login_required

def adminhome():
    return render_template('adminhome.html')

@app.route('/chat')
@login_required

def chat():
    return render_template('chat.html')

@app.route('/chatwithfarmers')
@login_required

def chatwithfarmers():
    qry = "select * from farmer"
    res = selectall(qry)
    return render_template('chatwithfarmers.html',val=res)



@app.route('/chat1',methods=['GET','POST'])
@login_required

def chat1():
    uid=request.args.get('uid')
    session['uidd']=uid
    qry=("select fname from farmer where lid=%s")
    val=str(uid)
    s1 = selectone(qry,val)
    fid=session['lid']
    session['idd'] = uid
    qry=("select * from chat where (fid=%s and tid=%s) or (fid=%s and tid=%s) order by date asc")
    val=(str(uid),str(fid),str(fid),str(uid))
    print(val)
    s = selectall2(qry,val)
    print(s)
    return render_template('chat.html',data=s,fname=s1[0],fr=str(uid))

@app.route('/chat11',methods=['GET','POST'])
@login_required

def chat11():
    uid = session['uidd']
    qry=("select fname from farmer where lid=%s")
```

```
    val=str(uid)
    s1 = selectone(qry,val)
    fid=session['lid']
    session['idd'] = uid
    qry=("select * from chat where (fid=%s and tid=%s) or (fid=%s and tid=%s) order by date asc")
    val=(str(uid),str(fid),str(fid),str(uid))
    print(val)
    s = selectall2(qry,val)
    print(s)
    return render_template('chat.html',data=s,fname=s1[0],fr=str(uid))
@app.route('/send',methods=['GET','POST'])
@login_required

def send():

      fid=session["lid"]
      print(fid)
      tid=session['idd']
      session['uidd']=tid
      print(tid)
      msg=request.form['textarea']
      qry=("insert into chat values(null,%s,%s,%s,curdate())")
      val=str(fid),str(tid),msg
      iud(qry,val)
      return '''<script>window.location='/chat11'</script>'''



@app.route('/fertilizer')
@login_required

def fertilizer():
    qry="SELECT * FROM `fertilizer`"
    res=selectall(qry)

    return render_template('fertilizer.html',val=res)

@app.route('/sendnotification')
@login_required

def sendnotification():

    return render_template('sendnotification.html')

@app.route('/sendingnotification',methods=['post'])
@login_required

def sendingnotification():
    noti=request.form['textarea']
    qry="insert into notification values(null,%s,curdate())"
    iud(qry,noti)
    return '''<script>alert("notification sent");window.location="/adminhome"</script>'''


@app.route('/viewfarmer')
@login_required

def viewfarmer():
    qry="select * from farmer"
    res=selectall(qry)
    return render_template('viewfarmer.html',val=res)

@app.route('/viewfeedback')
```

Appendix

```
@login_required

def viewfeedback():
   qry="SELECT `feedback`.*,`farmer`.* FROM `farmer` JOIN `feedback` ON `feedback`.`lid`=`farmer`.`lid`"
   res=selectall(qry)
   return render_template('viewfeedback.html',val=res)

@app.route('/deleteferti',methods=['get'])
@login_required

def deleteferti():
   id=request.args.get('id')
   qry="delete from fertilizer where fertid=%s"

   iud(qry,id)
   return '''<script>alert("deleted");window.location="/fertilizer"</script>'''



@app.route('/logout')
def logout():
   session.clear()
   return redirect('/')


app.run(debug=True
\\android.py
import os
from flask import *
from werkzeug.utils import secure_filename



from src.dbconnection import *
from src.myknn import prep

app=Flask(__name__)


@app.route("/logincode",methods=['post'])
def logincode():
   print(request.form)
   username=request.form['username']
   password=request.form['password']
   q="SELECT * FROM `login`WHERE `username`=%s AND`password`=%s"
   val=username,password
   res=selectone(q,val)
   print(res)
   if res is None:
      return jsonify({'task': 'unsuccessfull'})
   else:
      return jsonify({'task': 'success','lid':res[0],'type':res[3]})




@app.route("/farmer",methods=['post'])
def farmer():
   try:
      print(request.form)

      fname = request.form['fname']
```

Appendix

```
        lname = request.form['lname']
        phone = request.form['Phone']
        place = request.form['Place']
        pin = request.form['pin']
        post = request.form['post']
        email = request.form['Email']
        uname = request.form['username']
        passd = request.form['password']



        qry="INSERT INTO `login` VALUE(null,%s,%s,'farmer')"
        val=(uname,passd)
        id=iud(qry,val)
        qy="insert into farmer values( null,%s,%s,%s,%s,%s,%s,%s,%s)"
        val=(id,fname,lname,phone,place,pin,post,email)
        iud(qy,val)
        return jsonify({'task': 'success','id':id})
    except Exception as e:
        print(e)
        return jsonify({'task':str(e)})



@app.route("/viewnotification",methods=['post'])
def viewnotification():

    qry="SELECT * from `notification` "
    res = androidselectallnew(qry)
    print(res)
    return jsonify(res)







@app.route("/feedback",methods=['post'])
def feedback():
    id = request.form['id']
    fd = request.form['feedback']
    qry="INSERT INTO `feedback` VALUE(null,%s,curdate(),%s)"
    val=(fd,id)
    iud(qry,val)
    return jsonify({'task': 'success'})


@app.route("/viewfertilizer",methods=['post'])
def viewfertilizer():

    qry="SELECT * from fertilizer"
    res = androidselectallnew(qry)
    print(res)
    return jsonify(res)




@app.route('/img',methods=['post','get'])
def img():
    id=request.form['id']
```

Appendix

```
    img=request.files['files']
    fn=secure_filename(img.filename)
    img.save("static/chk/"+fn)
    res=prep("static/chk/"+fn)

    print(res[0])
    qry="insert into upload values(null,%s,%s,%s,curdate())"
    val=(id,str(res[0]),fn)
    iud(qry,val)
    if str(res[0])!='0':
        print("Healthy Leaves")
        qry = "insert into upload values(null,%s,'Healthy Leaves',%s,curdate())"
        val = (id, fn)
        iud(qry, val)
        return jsonify('Healthy Leaves')
    else:
        print("Healthy Leaves")
        qry = "insert into upload values(null,%s,'Black rot detected',%s,curdate())"
        val = (id,fn)
        iud(qry, val)
        print("Black rot detected")
        return jsonify('Black rot detected')
```

```
@app.route('/views',methods=['post'])
def views():
    lid=request.form['uid']
    print(lid)
    qry="SELECT * FROM login WHERE `lid` NOT IN (SELECT `lid` FROM `farmer`)"
    value=(lid)
    res = androidselectallnew(qry)
    return jsonify(res)


@app.route('/in_message2',methods=['post'])
def in_message():
    print(request.form)
    fromid = request.form['fid']
    print("fromid",fromid)

    toid = request.form['toid']
    print("toid",toid)

    message=request.form['msg']
    print("msg",message)
    qry = "INSERT INTO `chat` VALUES(NULL,%s,%s,%s,CURDATE())"
    value = (fromid, toid, message)
    print("pppppppppppppppppp")
    print(value)
    iud(qry, value)
    return jsonify(status='send')

@app.route('/view_message2',methods=['post'])
def view_message2():
    print("wwwwwwwwwwwwwwww")
    print(request.form)
    fromid=request.form['fid']
    print(fromid)
    toid=request.form['toid']
    print(toid)
```

Appendix

```
    lmid = request.form['lastmsgid']
    print("msgggggggggggggggggggggg"+lmid)
    sen_res = []
    # qry="SELECT * FROM chat WHERE (fromid=%s AND toid=%s) OR (fromid=%s AND toid=%s) ORDER BY DATE ASC"
    qry="SELECT `fid`,`message`,`date`,`chat_id` FROM `chat` WHERE `chat_id`>%s AND ((`tid`=%s AND `fid`=%s) OR (`tid`=%s
          AND `fid`=%s) ) ORDER BY chat_id ASC"

    val=(str(lmid),str(toid),str(fromid),str(fromid),str(toid))
    print("fffffffffffff",val)
    res = androidselectall(qry,val)
    print("resullllllllllll")
    print(res)
    if res is not None:
        return jsonify(status='ok', res1=res)
    else:
        return jsonify(status='not found')
app.run(port=5000,host="0.0.0.0")
```

Database Design

Insert the database desgin here (if any). A sample format is given below

Attribute Name	Datatype	length	Description
lid	Integer	11	Primary Key
username	Varchar	25	
password	varchar	25	
type	varchar	25	

Table A.1: Login

Attribute Name	Datatype	length	Description
nid	Integer	11	Primary Key
notification	Varchar	100	
date	varchar	25	

Table A.2: Notification

Attribute Name	Datatype	length	Description
fid	Integer	11	Primary Key
lid	Integer	11	
fname	varchar	25	
lname	varchar	25	
phone	biginteger	20	
place	varchar	25	
pin	integer	11	
post	varchar	25	
email	varchar	50	

Table A.3: Farmer

Attribute Name	Datatype	length	Description
chatid	Integer	11	Primary Key
fid	integer	11	
tid	integer	11	
message	varchar	50	
date	date		

Table A.4: Chat

Attribute Name	Datatype	length	Description
feedid	Integer	11	Primary Key
feedback	varchar	100	
date	date		
lid	int	11	

Table A.5: Feedback

Attribute Name	Datatype	length	Description
fertid	Integer	11	Primary Key
feetname	varchar	25	
users	varchar	50	
description	varchar	50	

Table A.6: Fertilizer

Attribute Name	Datatype	length	Description
uploadid	Integer	11	Primary Key
userid	int	11	
result	varchar	50	
image	varchar	50	
date	date		

Table A.7: Upload

DaTaflow Diagram

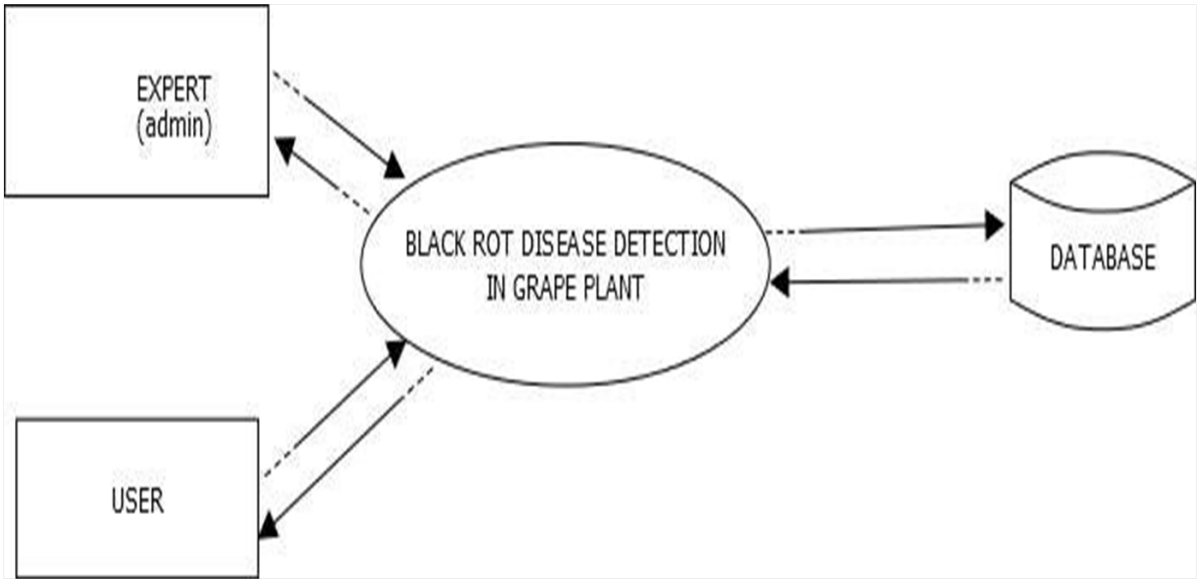

Figure A.1: Level 0

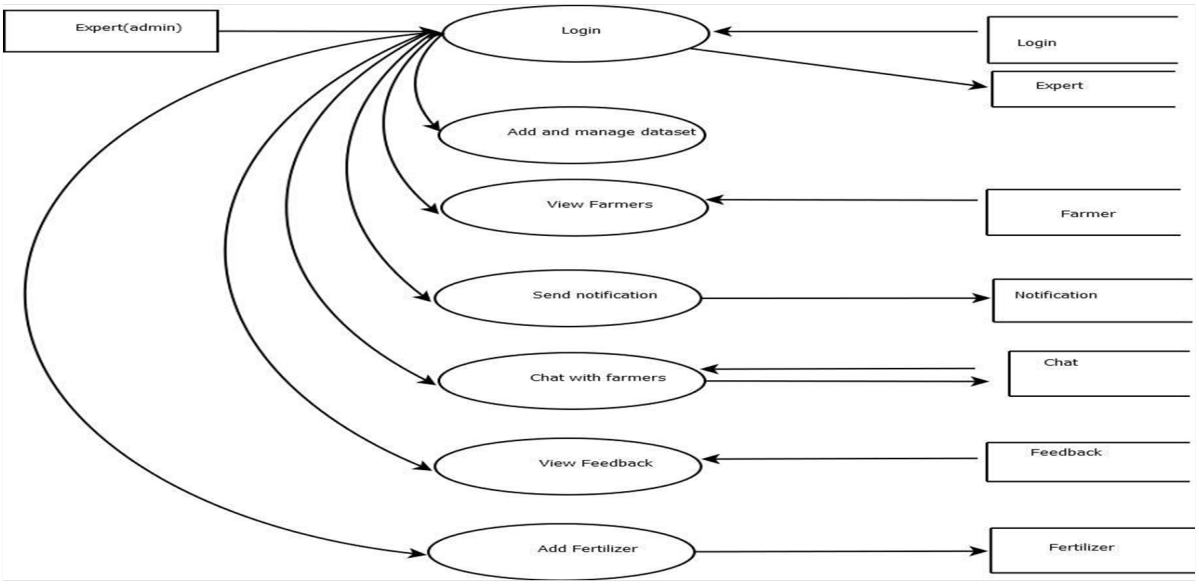

Figure A.2: Level 1

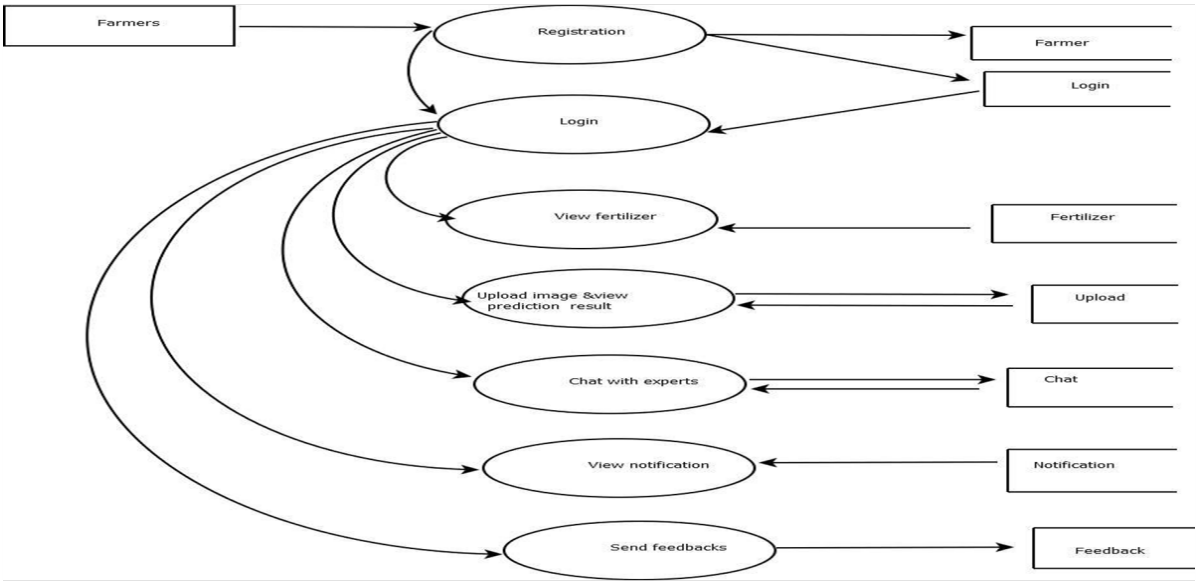

Figure A.3: Level 2